

# **Whitepaper** ***froglogic* Tq**

**Version 1.0**

**Copyright © *froglogic* 2003.**

**All rights reserved.**

## Abstract

Tq is a lightweight C++ library for programming mixed Tk and Qt applications. It is particularly useful for incrementally porting Tcl/Tk applications to C++/Qt. It also includes tqsh, a wish replacement that supports both Tk and Qt.

## Introduction

*froglogic* Tq is a library which makes it possible to mix Tcl/Tk and Qt code in a single application. Tq's main purpose is to ease the migration of Tcl/Tk applications to Qt applications. Tq is also useful if you want to use a Tk component in a Qt application.

Tq's key features are:

- Event Loop Integration
- Modal Dialog Behavior
- Embedding Tk Windows in Qt Widgets
- Clipboard and Color Unification
- Wish Replacement tqsh
- Documentation
- Using Qt From Tcl

The following sections present Tq's features in more detail.

## Event Loop Integration

The most important feature of a GUI toolkit migration library is event loop integration. This means that the application's event loop processes and dispatches both toolkits' events; in Tq's case this means both Qt and Tcl/Tk events.

Tq implements a custom event loop that hooks into Qt's central event dispatching routine and which also handles Tcl/Tk events.

This ensures that Tcl events are always processed correctly. To make sure that Qt events are also always processed correctly, Tq hooks into the Tcl/Tk event mechanism to ensure that Qt events are also processed when Tcl/Tk enters a local event loop: for example when opening a

modal Tk dialog.

Tq offers a convenience function, `TqMain()`, to simplify the initialization step of the mixed event loop. This function is analogous to Tk's `Tk_Main()`. It initializes Qt, Tcl and Tk and then enters the mixed event loop. The function returns after the main event loop exits.

The `TqApplication` is provided for users which need more control over the initialization, for example, to create their own `QApplication` subclass. The `TqApplication` class is used internally by `TqMain()`. The simple case for initializing a mixed Tcl/Tk/Qt application looks like this:

```
static int init_proc( Tcl_Interp *ip )
{
    MainWindow *mw = new MainWindow( ip, 0, "mainwindow" );
    mw->show();
    QObject::connect( qApp, SIGNAL( lastWindowClosed() ),
                     qApp, SLOT( quit() ) );
    return TCL_OK;
}

int main( int argc, char **argv )
{
    QString code = "[some tcl init code]";
    return TqMain( argc, argv, code,
                  Tq::PrimaryQt, &init_proc );
}
```

This code snippet initializes both toolkits, executes `init_proc()` and then evaluates the specified Tcl code. `Tq::PrimaryQt` specifies, that this application is primarily a Qt application and that exiting the Qt application is the determining factor for exiting the entire application.

If you need to instantiate our own `QApplication` subclass, you can use the following code:

```
class MyTqApp : public TqApplication
{
```

```
public:
    ....
protected:
    virtual QApplication *createQApplication( int argc,
                                             char **argv );
};

QApplication *MyTqApp::createQApplication( int argc,
                                           char **argv )
{
    return new MyQApplication( argc, argv );
}

....

int main( int argc, char **argv )
{
    MyTqApp tqapp( argc, argv, Tq::PrimaryTk );
    tqapp.setCode( "[some tcl code]" );
    tqapp.setAppInitProc( &init_proc );
    return tqapp.exec();
}
```

In this case `MyTqApp::createQApplication()` will be called by `exec()`, resulting in `MyQApplication` being instantiated.

## Modal Dialog Behavior

Both Qt and Tk support modality. This means that if the toolkit opens a modal dialog, a local event loop is entered and user input to all other windows of the toolkit is blocked. In a mixed Qt/Tk application such behavior is desirable across toolkit boundaries as well. For example, a modal Qt dialog should also block user input from the application's Tk windows and vice versa.

Tq transparently takes care of this, ensuring correct modal behavior for both Qt and Tk dialogs in a mixed toolkit application.

## Embedding Tk Windows into Qt Widgets

If the application's main window is a `QMainWindow` (or `QMainWindow` subclass), it is often desirable to be able to embed Tk windows into Qt widgets. Tq makes this possible with its `TqWidget` class. When creating a `TqWidget` a Tk window of the same name is created. This window can be accessed like any other Tk window from Tcl. And since `TqWidget` is also a `QWidget` subclass, it can be used like any other `QWidget` in Qt. This means for example, that the widget can be laid out using Qt's layout system. We can create a `TqWidget` in C++ like this:

```
TqWidget *w = new TqWidget( interp, mainwindow, ".w" );
```

After that it can be used in Tcl as usual. For example we would populate it with a Tk button like this:

```
pack [button .w.b -text "Push Me"]
```

## Clipboard and Style Unification

Both Qt and Tk use X11 selections for clipboard handling. This means that text can be copied & pasted between Qt and Tk widgets in a single application.

Tk doesn't offer a style system such as Qt's to influence the look of widgets without having to modify the widget implementation itself. Consequently it's possible to spot the visual difference between Qt and Tk widgets in a mixed Qt/Tk application. To make both toolkits look as similar as possible, Tq applies the palette and font used by Qt to Tk so that the same colors and fonts are used in Qt and Tk widgets.

In addition Tq makes sure that Qt and Tk share the same X11 color map and visual to save resources.

## Wish Replacement tqsh

Many Tcl/Tk applications are executed via the wish shell. This shell enters a Tcl/Tk event loop and evaluates Tcl code.

Tq comes with a command line tool called tqsh. This can replace wish for mixed Tcl/Tk/Qt application. Tqsh enters a Tq event loop and in a similar way to wish, tqsh offers an interactive console with syntax highlighting if no script is specified on the command line.

The interactive console is also available via the TqConsole class and can be integrated into Tq applications.

## Documentation

Tq comes with reference documentation in HTML format. The documentation covers the complete API and explains the concepts behind Tq. In addition a tutorial shows how to migrate parts of a pure Tcl/Tk application to Qt. Different examples show different usages of Tq.

## Using Qt from Tcl

Tq doesn't come with Tcl bindings for Qt. While this means that Qt cannot be used from Tcl directly, usually it is not necessary to have full access to Qt from Tcl when migrating an application. In most cases only certain functionality, like opening a dialog implemented in Qt might be required. This can be achieved by adding such functionality as commands to the Tcl interpreter. Here is a small example, which adds a command to open the QFileDialog (for simplicity, all sanity checks in the code have been omitted):

```
int qGetSaveFileNameCmd( ClientData, Tcl_Interp *ip,
                        int objc, Tcl_Obj * const objv[] )
{
    QString f = QFileDialog::getSaveFileName();
    if ( !f.isNull() ) {
        char *n = (char*)f.ascii();
        Tcl_SetObjResult( ip, Tcl_NewStringObj( n, -1 ) );
    }
    return TCL_OK;
}
```

```
Tcl_CreateObjCommand( ip, "qGetSaveFileName",  
                    &qGetSaveFileNameCmd, 0, 0 );
```

The following Tcl code opens the `QFileDialog` and prints out the chosen file name:

```
set fn [qGetOpenFileName]  
puts $fn
```

Tcl bindings might be added in the future, but even without this, it is straightforward to make Qt functionality available to Tcl.

## **Conclusion**

Tq offers everything necessary to make the migration from Tcl/Tk to Qt as painless and simple as possible. The code needed to make use of Tq is very small in comparison with an entire application's code, and the performance impact of Tq is negligible. This makes Tq a very useful tool for migrating Tcl/Tk applications to Qt or in general to mix those two toolkits in a single application.

If you have any further questions regarding Tq, please contact us at [tq@froglogic.com](mailto:tq@froglogic.com). You can also visit [www.froglogic.com](http://www.froglogic.com) for further information about Tq and other *froglogic* products and services.